Your search matched **7** of **1131693** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering a new one in the text box.

backtracking and debugging Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **Backtracking and re-execution in the automatic debugging of parallelized programs**
Matthews, G.; Hood, R.; Johnson, S.; Leggett, P.;
High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on , 23-26 July 2002
Pages:150 - 160

[Abstract] [PDF Full-Text (321 KB)] **IEEE CNF**

---

2 **An execution-backtracking approach to debugging**
Agrawal, H.; De Millo, R.A.; Spafford, E.H.;
Software, IEEE , Volume: 8 , Issue: 3 , May 1991
Pages:21 - 26

[Abstract] [PDF Full-Text (528 KB)] **IEEE JNL**

---

3 **Visual programming with graph rewriting systems**
Schurr, A.; Winter, A.; Zundorf, A.;
Visual Languages, Proceedings., 11th IEEE International Symposium on , 5-9 Sept. 1995
Pages:326 - 333

[Abstract] [PDF Full-Text (928 KB)] **IEEE CNF**

---

4 **Cognitive activities and support in debugging**
Byung-Do Yoon; Garcia, O.N.;
Human Interaction with Complex Systems, 1998. Proceedings., Fourth Annual Symposium on , 22-25 March 1998
Pages:160 - 169

[Abstract] [PDF Full-Text (688 KB)] **IEEE CNF**

---

5 **PELAS-program error-locating assistant system**
Korel, B.;

Software Engineering, IEEE Transactions on , Volume: 14 , Issue: 9 , Sept. 1988
Pages:1253 - 1260

[Abstract] [PDF Full-Text (644 KB)] IEEE JNL

---

6 **A precise demand-driven definition-use chaining algorithm**
Hajnal, A.; Forgacs, I.;
Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on , 11-13 March 2002
Pages:77 - 86

[Abstract] [PDF Full-Text (379 KB)] IEEE CNF

---

7 **Reversible functional simulation for digital system design**
Jennings, G.;
Custom Integrated Circuits Conference, 1991., Proceedings of the IEEE 1991 , 12-15 May 1991
Pages:8.2/1 - 8.2/4

[Abstract] [PDF Full-Text (316 KB)] IEEE CNF

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search | Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting | No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

Terms used
backtracking history transformation tree result XSLT

Found 1 of 10,202 searched out of 10,202.

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 1 of 1

Relevance scale

1 XML Applications: An incremental XSLT transformation processor for XML document manipulation

Lionel Villard, Nabil Layaïda

May 2002 **Proceedings of the eleventh international conference on World Wide Web**

Full text available: pdf(486.95 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we present an incremental transformation framework called incXSLT. This framework has been experimented for the XSLT language defined at the World Wide Web Consortium. For the currently available tools, designing the XML content and the transformation sheets is an inefficient, a tedious and an error prone experience. Incremental transformation processors such as incXSLT represent a better alternative to help in the design of both the content and the transformation sheets. We belie ...

Keywords: XML, XSLT, authoring tools, incremental transformations

Results 1 - 1 of 1

ACM PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ◉ The ACM Digital Library ○ The Guide

debugging SEARCH

THE ACM DIGITAL LIBRARY

Advanced Search

Search Tips

Enter words, phrases or names below. Surround phrases or full names with double quotation marks.

Search within Results: **10,202** found

debugging

Clear result set

SEARCH

Desired Results:

must have **all** of the words or phrases

backtracking history transformation tree result

must have **any** of the words or phrases

must have **none** of the words or phrases

Name or Affiliation:

Authored by: ◉ all ○ any ○ none

Edited by: ◉ all ○ any ○ none

Reviewed by: ◉ all ○ any ○ none

Only search in:*

○ Title ○ Abstract ○ Review ◉ All Information

SEARCH

*Searches will be performed on all available information, including full text where available, unless specified above.

ISBN / ISSN: ◉ Exact ○ Expand

DOI: ◉ Exact ○ Expand

SEARCH

Published:

By: ◉ all ○ any ○ none

In: ◉ all ○ any ○ none

Since: Month Year

Before: Month Year

As: Any type of publication

Conference Proceeding:

Sponsored By:

Conference Location:

Conference Year: yyyy

SEARCH

Classification: (CCS) ☐ **Primary Only**

Classified as: ◉ all ○ any ○ none

Results must have accessible:

☐ Full Text ☐ Abstract ☐ Review

Subject Descriptor: ◉ all ○ any ○ none

Keyword Assigned: ◉ all ○ any ○ none

SEARCH

Terms used **backtracking history transformation tree result**

Found **64** of **10,202** searched out of **10,202.**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 64

Result page: **1** 2 3 4 next

Relevance scale

1 Computing curricula 2001

September 2001 **Journal on Educational Resources in Computing (JERIC)**

Full text available: pdf(613.63 KB) html(2.78 KB) Additional Information: full citation, references, citings, index terms

2 Program Transformation Systems

H. Partsch, R. Steinbrüggen

September 1983 **ACM Computing Surveys (CSUR)**, Volume 15 Issue 3

Full text available: pdf(3.00 MB) Additional Information: full citation, references, citings, index terms

3 Transformational derivation of programs using the focus system

Uday S. Reddy

November 1988 **Proceedings of the third ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments**, Volume 13 , 24 Issue 5 , 2

Full text available: pdf(1.30 MB) Additional Information: full citation, references, citings, index terms

4 XML Applications: An incremental XSLT transformation processor for XML document manipulation

Lionel Villard, Nabil Layaïda

May 2002 **Proceedings of the eleventh international conference on World Wide Web**

Full text available: pdf(486.95 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we present an incremental transformation framework called incXSLT. This framework has been experimented for the XSLT language defined at the World Wide Web Consortium. For the currently available tools, designing the XML content and the transformation sheets is an inefficient, a tedious and an error prone experience. Incremental transformation processors such as incXSLT represent a better alternative to help in the design of both the content and the transformation sheets. We belie ...

Keywords: XML, XSLT, authoring tools, incremental transformations

5 Technical papers: software architecture: Advanced control flows for flexible graphical

user interfaces: or, growing GUIs on trees or, bookmarking GUIs
Paul T. Graunke, Shriram Krishnamurthi
May 2002 **Proceedings of the 24th International Conference on Software Engineering**

Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, index terms

Web and GUI programs represent two extremely common and popular modes of human-computer interaction. Many GUI programs share the Web's notion of *browsing* through data- and decision-trees. This paper compares the user's browsing power in the two cases and illustrates that many GUI programs fall short of the Web's power to clone windows and bookmark applications. It identifies a key implementation problem that GUI programs must overcome to provide this power. It then describes a theoretical ...

6 The family of concurrent logic programming languages
Ehud Shapiro
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

7 Parsing and compiling using Prolog
Jacques Cohen, Timothy J. Hickey
March 1987 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 9 Issue 2

Full text available: pdf(2.83 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper presents the material needed for exposing the reader to the advantages of using Prolog as a language for describing succinctly most of the algorithms needed in prototyping and implementing compilers or producing tools that facilitate this task. The available published material on the subject describes one particular approach in implementing compilers using Prolog. It consists of coupling actions to recursive descent parsers to produce syntax-trees which are subsequently utilized ...

8 Fast detection of communication patterns in distributed executions
Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

9 An optimizing compiler for lexically scoped LISP
Rodney A. Brooks, Richard P. Gabriel, Guy L. Steele
June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction**, Volume 17 Issue 6

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

We are developing an optimizing compiler for a dialect of the LISP language. The current target architecture is the S-I, a multiprocessing supercomputer designed at Lawrence

Livermore National Laboratory. While LISP is usually thought of as a language primarily for symbolic processing and list manipulation, this compiler is also intended to compete with the S-1 PASCAL and FORTRAN compilers for quality of compiled numerical code. The S-1 is designed for extremely high-speed signal processing ...

10 Demonic memory for process histories

P. R. Wilson, T. G. Moher

June 1989 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation**, Volume 24 Issue 7

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

Demonic memory is a form of reconstructive memory for process histories. As a process executes, its states are regularly checkpointed, generating a history of the process at low time resolution. Following the initial generation, any prior state of the process can be reconstructed by starting from a checkpointed state and re-executing the process up through the desired state, thereby exploiting the redundancy between the states of a process and the description of that process (i.e., a comput ...

11 The FINITE STRING Newsletter: Abstracts of current literature

Computational Linguistics Staff

January 1987 **Computational Linguistics**, Volume 13 Issue 1-2

Full text available: pdf(6.15 MB) Additional Information: full citation

Publisher Site

12 Special issue on prototypes of deductive database systems: The aditi deductive database system

Jayen Vaghani, Kotagiri Ramamohanarao, David B. Kemp, Zoltan Somogyi, Peter J. Stuckey, Tim S. Leask, James Harland

April 1994 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 3 Issue 2

Full text available: pdf(2.67 MB) Additional Information: full citation, abstract, references, citings

Deductive databases generalize relational databases by providing support for recursive views and non-atomic data. Aditi is a deductive system based on the client-server model; it is inherently multi-user and capable of exploiting parallelism on shared-memory multiprocessors. The back-end uses relational technology for efficiency in the management of disk-based data and uses optimization algorithms especially developed for the bottom-up evaluation of logical queries involving recursion. The front ...

Keywords: implementation, logic, multi-user, parallelism, relational database

13 Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach

André Baresel, David Binkley, Mark Harman, Bogdan Korel

July 2004 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 29 Issue 4

Full text available: pdf(241.32 KB) Additional Information: full citation, abstract, references, index terms

Evolutionary testing is an effective technique for automatically generating good quality test data. However, for structural testing, the technique degenerates to random testing in the presence of flag variables, which also present problems for other automated test data generation techniques. Previous work on the flag problem does not address flags assigned in loops.This paper introduces a testability transformation that transforms programs with loop--assigned flags so that existing genetic appro ...

Keywords: empirical evaluation, evolutionary testing, flags, testability transformation

14 Describing Prolog by its interpretation and compilation
Jacques Cohen
December 1985 **Communications of the ACM**, Volume 28 Issue 12

Full text available: pdf(1.50 MB) Additional Information: full citation, abstract, references, citings, index terms

Since its conception, Prolog has followed a developmental course similar to the early evolution of LISP. Although the version of Prolog described here typifies that currently in use, it should be considered within the framework of language evolution.

15 Generating test cases for real-time systems from logic specifications
Dino Mandrioli, Sandro Morasca, Angelo Morzenti
November 1995 **ACM Transactions on Computer Systems (TOCS)**, Volume 13 Issue 4

Full text available: pdf(2.31 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We address the problem of automated derivation of functional test cases for real-time systems, by introducing techniques for generating test cases from formal specifications written in TRIO, a language that extends classical temporal logic to deal explicitly with time measures. We describe an interactive tool that has been built to implement these techniques, based on interpretation algorithms of the TRIO language. Several heuristic criteria are suggested to reduce drastically the size of t ...

16 Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science
William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, David M. Young
March 1968 **Communications of the ACM**, Volume 11 Issue 3

Full text available: pdf(6.63 MB) Additional Information: full citation, references, citings

Keywords: computer science academic programs, computer science bibliographies, computer science courses, computer science curriculum, computer science education, computer science graduate programs, computer science undergraduate programs

17 Special issue on prototypes of deductive database systems: The CORAL deductive system
Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, Praveen Seshadri
April 1994 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 3 Issue 2

Full text available: pdf(3.03 MB) Additional Information: full citation, abstract, references, citings

CORAL is a deductive system that supports a rich declarative language, and an interface to C++, which allows for a combination of declarative and imperative programming. A CORAL declarative program can be organized as a collection of interacting modules. CORAL supports a wide range of evaluation strategies, and automatically chooses an efficient strategy for each module in the program. Users can guide query optimization by selecting from a wide range of control choices. The CORAL system provides ...

Keywords: deductive database, logic programming system, query language

18 Launching the new era
Kazuhiro Fuchi, Robert Kowalski, Koichi Furukawa, Kazunori Ueda, Ken Kahn, Takashi

Chikayama, Evan Tick
March 1993 **Communications of the ACM**, Volume 36 Issue 3
Full text available: pdf(3.45 MB) Additional Information: full citation, references, index terms, review

19 A history of the SNOBOL programming languages
Ralph E. Griswold
January 1978 **ACM SIGPLAN Notices , The first ACM SIGPLAN conference on History of programming languages**, Volume 13 Issue 8
Full text available: pdf(3.56 MB) Additional Information: full citation, abstract, references, index terms

Development of the SNOBOL language began in 1962. It was followed by SNOBOL2, SNOBOL3, and SNOBOL4. Except for SNOBOL2 and SNOBOL3 (which were closely related), the others differ substantially and hence are more properly considered separate languages than versions of one language. In this paper historical emphasis is placed on the original language, SNOBOL, although important aspects of the subsequent languages are covered.

20 Strategic directions in constraint programming
Pascal Van Hentenryck, Vijay Saraswat
December 1996 **ACM Computing Surveys (CSUR)**, Volume 28 Issue 4
Full text available: pdf(402.08 KB) Additional Information: full citation, references, citings, index terms

Results 1 - 20 of 64 Result page: **1** 2 3 4 next

Terms used **backtracking history transformation tree result** Found **64** of **10,202**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 21 - 40 of 64 Result page: previous 1 **2** 3 4 next

Relevance scale

21 HydroJ: object-oriented pattern matching for evolvable distributed systems

Keunwoo Lee, Anthony LaMarca, Craig Chambers

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11

Full text available: pdf(311.06 KB) Additional Information: full citation, abstract, references, index terms

In an evolving software system, components must be able to change independently while remaining compatible with their peers. One obstacle to independent evolution is the *brittle parameter problem*: the ability of two components to communicate can depend on a number of *inessential* details of the types, structure, and/or contents of the values communicated. If these details change, then the components can no longer communicate, even if the *essential* parts of the message remain ...

Keywords: HydroJ, XML, distributed systems, dynamic dispatch, object-oriented programming, pattern matching, semi-structured data, software evolution, ubiquitous computing

22 A view of the origins and development of Prolog

Jacques Cohen

January 1988 **Communications of the ACM**, Volume 31 Issue 1

Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

Dealing with failure is easy: Work hard to improve. Success is also easy to handle: You've solved the wrong problem. Work hard to improve.

23 Resourceful systems for fault tolerance, reliability, and safety

Russell J. Abbott

March 1990 **ACM Computing Surveys (CSUR)**, Volume 22 Issue 1

Full text available: pdf(3.36 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Above all, it is vital to recognize that completely guaranteed behavior is impossible and that there are inherent risks in relying on computer systems in critical environments. The unforeseen consequences are often the most disastrous [Neumann 1986]. Section 1 of this survey reviews the current state of the art of system reliability, safety, and fault tolerance. The emphasis is on the contribution of software to these areas. Section 2 reviews current approaches to software fault ...

24 In black and white: an integrated approach to class-level testing of object-oriented

programs
Huo Yan Chen, T. H. Tse, F. T. Chan, T. Y. Chen
July 1998 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 7 Issue 3

Full text available: pdf(284.99 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Because of the growing importance of object-oriented programming, a number of testing strategies have been proposed. They are based either on pure black-box or white-box techniques. We propose in this article a methodology to integrate the black- and white-box techniques. The black-box technique is used to select test cases. The white-box technique is mainly applied to determine whether two objects resulting from the program execution of a test care are observationally equivalent. It is als ...

Keywords: abstract data types, algebraic specification, object-oriented programming, observational equivalence, software-testing methodologies

25 Cost effective dynamic program slicing
Xiangyu Zhang, Rajiv Gupta
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6

Full text available: pdf(234.12 KB) Additional Information: full citation, abstract, references, index terms

Although dynamic program slicing was first introduced to aid in user level debugging, applications aimed at improving software quality, reliability, security, and performance have since been identified as candidates for using dynamic slicing. However, the dynamic dependence graph constructed to compute dynamic slices can easily cause slicing algorithms to run out of memory for realistic program runs. In this paper we present the design and evaluation of a cost effective dynamic program slicing a ...

Keywords: debugging, dynamic dependence graph, testing

26 The treatment of state in optimistic systems
David Bruce
July 1995 **ACM SIGSIM Simulation Digest , Proceedings of the ninth workshop on Parallel and distributed simulation**, Volume 25 Issue 1

Full text available: pdf(1.59 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Optimistic computation methods typically save copies of objects' state information, so that they can recover from erroneous "over-optimistic" computations. Such state saving is generally time and space consuming, and can be rather complicated both to implement and to use.I show how the data structure community's theory of persistence can be used not only to analyse and explain the treatment of state in optimistic systems, but also as a simple yet general mechanism for ...

Keywords: optimistic methods, parallel discrete event simulation, persistent data structures, state saving

27 A slicing-based approach for locating type errors
F. Tip, T. B. Dinesh
January 2001 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 10 Issue 1

Full text available: pdf(443.36 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The effectiveness of a type-checking tool strongly depends on the accuracy of the positional information that is associated with type errors. We present an approach where the location

associated with an error message e is defined as a slice Pe of the program P being type-checked. We show that this approach yields highly accurate positional information: Pe is a progr ...

Keywords: abstract interpretation, program slicing, semantics-based tool generation, static semantics, type-checking

28 The Pan language-based editing system

Robert A. Ballance, Susan L. Graham, Michael L. Van De Vanter

January 1992 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 1 Issue 1

Full text available: pdf(2.43 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Powerful editing systems for developing complex software documents are difficult to engineer. Besides requiring efficient incremental algorithms and complex data structures, such editors must accommodate flexible editing styles, provide a consistent, coherent, and powerful user interface, support individual variations and projectwide configurations, maintain a sharable database of information concerning the documents being edited, and integrate smoothly with the other tools in the environme ...

Keywords: Ladle, Pan, coherent user interfaces, colander, contextual constraint, extension facilities, grammatical abstraction, interactive programming environment, logic programming, logical constraint grammar, reason maintenance, syntax-recognizing editor, tolerance for errors and anomalies

29 ITiCSE 2002 working group report: Exploring the role of visualization and engagement in computer science education

Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, J. Ángel Velázquez-Iturbide

June 2002 **ACM SIGCSE Bulletin , Working group reports from ITiCSE on Innovation and technology in computer science education**, Volume 35 Issue 2

Full text available: pdf(414.24 KB) Additional Information: full citation, abstract, references, citings

Visualization technology can be used to graphically illustrate various concepts in computer science. We argue that such technology, no matter how well it is designed, is of little educational value unless it engages learners in an active learning activity. Drawing on a review of experimental studies of visualization effectiveness, we motivate this position against the backdrop of current attitudes and best practices with respect to visualization use. We suggest a new taxonomy of learner engageme ...

30 Programming languages for distributed computing systems

Henri E. Bal, Jennifer G. Steiner, Andrew S. Tanenbaum

September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available: pdf(6.50 MB) Additional Information: full citation, abstract, references, citings, index terms, review

When distributed systems first appeared, they were programmed in traditional sequential languages, usually with the addition of a few library procedures for sending and receiving messages. As distributed applications became more commonplace and more sophisticated, this ad hoc approach became less satisfactory. Researchers all over the world began designing new programming languages specifically for implementing distributed applications. These languages and their history, their underlying pr ...

31 Translator writing systems

Jerome Feldman, David Gries

February 1968 **Communications of the ACM**, Volume 11 Issue 2

Full text available: pdf(4.47 MB) Additional Information: full citation, abstract, references, citings

A critical review of recent efforts to automate the writing of translators of programming languages is presented. The formal study of syntax and its application to translator writing are discussed in Section II. Various approaches to automating the postsyntactic (semantic) aspects of translator writing are discussed in Section III, and several related topics in Section IV.

Keywords: compiler compiler-compiler, generator, macroprocessor, meta-assembler, metacompiler, parser, semantics, syntactic analysis, syntax, syntax-directed, translator, translator writing system

32 Efficient detection of determinacy races in Cilk programs

Mingdong Feng, Charles E. Leiserson

June 1997 **Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures**

Full text available: pdf(1.46 MB) Additional Information: full citation, references, citings, index terms

33 Special issues on machine translation: Automated translation at Grenoble University

Bernard Vauquois, Christian Boitet

January 1985 **Computational Linguistics**, Volume 11 Issue 1

Full text available: pdf(1.01 MB) Publisher Site Additional Information: full citation, references, citings

34 Recipes for geometry and numerical analysis - Part I: an empirical study

D. P. Dobkin, D. Silver

January 1988 **Proceedings of the fourth annual symposium on Computational geometry**

Full text available: pdf(1.00 MB) Additional Information: full citation, abstract, references, citings, index terms

Geometric computations, like all numerical procedures, are extremely prone to roundoff error. However, virtually none of the numerical analysis literature directly applies to geometric calculations. Even for line intersection, the most basic geometric operation, there is no robust and efficient algorithm. Compounding the difficulties, many geometric algorithms perform iterations of calculations reusing previously computed data. In this paper, we explore some of the main issues in geometric ...

35 Constraints

Guy Lewis Steele, Gerald Jay Sussman

May 1979 **ACM SIGAPL APL Quote Quad , Proceedings of the international conference on APL: part 1**, Volume 9 Issue 4

Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

We present an interactive system organized around networks of constraints rather than the programs which manipulate them. We describe a language of hierarchical constraint networks. We describe one method of deriving useful consequences of a set of constraints which we call propagation. Dependency analysis is used to spot and track down inconsistent subsets of a constraint set. Propagation of constraints is most flexible and useful when coupled with the ability to perform symbolic manipulat ...

Keywords: Constraints, Declarative languages, Dependencies, Propagation of constraints, Responsible programs, Sketchpad

36 An introduction to partial evaluation
Neil D. Jones
September 1996 **ACM Computing Surveys (CSUR)**, Volume 28 Issue 3

Full text available: pdf(546.94 KB) Additional Information: full citation, abstract, references, citings, index terms

Partial evaluation provides a unifying paradigm for a broad spectrum of work in program optimization compiling interpretation and the generation of automatic program generators [Bjørner et al. 1987; Ershov 1992; and Jones et al. 1993]. It is a program optimization technique, perhaps better called program specialization, closely related to but different from Jørring and Scherlis' staging transformations [1986]. It emphasizes, in comparison with ...

Keywords: compiler generators, compilers, interpreters, partial evaluation, program specialization

37 Improvements to graph coloring register allocation
Preston Briggs, Keith D. Cooper, Linda Torczon
May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 3

Full text available: pdf(2.00 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We describe two improvements to Chaitin-style graph coloring register allocators. The first, optimistic coloring, uses a stronger heuristic to find a k-coloring for the interference graph. The second extends Chaitin's treatment of rematerialization to handle a larger class of values. These techniques are complementary. Optimistic coloring decreases the number of procedures that require spill code and reduces the amount of spill code when sp ...

Keywords: code generation, graph coloring, register allocation

38 A generator for language-specific debugging systems
R. Bahlke, B. Moritz, G. Snelting
July 1987 **ACM SIGPLAN Notices , Papers of the Symposium on Interpreters and interpretive techniques**, Volume 22 Issue 7

Full text available: pdf(583.68 KB) Additional Information: full citation, abstract, citings, index terms

We present a system which generates interactive high-level debugging systems from formal language definitions. The language definer has to specify a denotational semantics augmented with a formal description of the language specific debugging facilities. The generated debugger offers the traditional features such as tracing programs, setting breakpoints, displaying variables etc; interaction with the user is always on language level rather than on machine level. The concept has been implemented ...

39 An environment for logic programming
Nissim Francez, Shalom Goldenberg, Ron Y. Pinter, Michael Tiomkin, Shalom Tsur
June 1985 **Proceedings of the ACM SIGPLAN 85 symposium on Language issues in programming environments**, Volume 20 , 18 Issue 7 , 6

Full text available: pdf(940.71 KB) Additional Information: full citation, abstract, references, index terms

We describe a programming environment for Prolog, a common logic programming language. The services offered by our system assist a Prolog user in the tasks of composing, editing, and storing logic (rule-based) programs, as well as in the control of their execution for debugging purposes. In order to facilitate effective debugging of Prolog programs, we propose a new model of computation that can handle both pure Prolog and impure (side-effect causing) Prolog oper ...

40 Using object-orientation to implement logic programming
Phillip Cox
February 1990 **Proceedings of the 1990 ACM SIGSMALL/PC symposium on Small**

systems

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, index terms

Since their introduction, microcomputers have undergone a revolution in which the standard text-based input and output facilities have been replaced by powerful graphics and pointing devices. As a result, most applications are now driven by easy-to-use pictorial interfaces. Consequently, microcomputers are now mainly used by people who are not computer professionals and expect increasingly sophisticated and powerful applications. The resulting pressure on the software development industry h ...

Results 21 - 40 of 64 Result page: previous 1 **2** 3 4 next

Terms used **backtracking history transformation tree result**

Found **64** of **10,202**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 41 - 60 of 64

Result page: previous 1 2 **3** 4 next

Relevance scale

41 Verification techniques for cache coherence protocols

Fong Pong, Michel Dubois

March 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 1

Full text available: pdf(1.25 MB)

Additional Information: full citation, abstract, references, citings, index terms

In this article we present a comprehensive survey of various approaches for the verification of cache coherence protocols based on state enumeration, (symbolic model checking, and symbolic state models. Since these techniques search the state space of the protocol exhaustively, the amount of memory required to manipulate that state information and the verification time grow very fast with the number of processors and the complexity of the protocol mechanism ...

Keywords: cache coherence, finite state machine, protocol verification, shared-memory multiprocessors, state representation and expansion

42 Towards monolingual programming environments

Jan Heering, Paul Klint

April 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 7 Issue 2

Full text available: pdf(2.66 MB)

Additional Information: full citation, abstract, references, citings, index terms, review

Most programming environments are much too complex. One way of simplifying them is to reduce the number of mode-dependent languages the user has to be familiar with. As a first step towards this end, the feasibility of unified command/programming/debugging languages, and the concepts on which such languages have to be based, are investigated. The unification process is accomplished in two phases. First, a unified command/programming framework is defined and, second, this framework is extend ...

43 Program abstraction and instantiation

Nachum Dershowitz

July 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 7 Issue 3

Full text available: pdf(2.18 MB)

Additional Information: full citation, abstract, references, citings, index terms, review

Our goal is to develop formal methods for abstracting a given set of programs into a program schema and for instantiating a given schema to satisfy concrete specifications. Abstraction and instantiation are two important phases in software development which allow programmers to apply knowledge learned in the solutions of past problems when faced with new situations. For example, from two programs using a linear (or binary) search technique,

an abstract schema can be derived that embodies th ...

44 A new notation for arrows

Ross Paterson

October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming**, Volume 36 Issue 10

Full text available: pdf(211.35 KB) Additional Information: full citation, abstract, references, citings, index terms

The categorical notion of monad, used by Moggi to structure denotational descriptions, has proved to be a powerful tool for structuring combinator libraries. Moreover, the monadic programming style provides a convenient syntax for many kinds of computation, so that each library defines a new sublanguage. Recently, several workers have proposed a generalization of monads, called variously "arrows" or Freyd-categories. The extra generality promises to increase the power, expressiveness and efficie ...

45 A software engineering perspective on algorithmics

Karsten Weihe

March 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 1

Full text available: pdf(1.62 MB) Additional Information: full citation, abstract, references, index terms, review

An algorithm component is an implementation of an algorithm which is not intended to be a stand-alone module, but to perform a specific task within a large software package or even within several distinct software packages. Therefore, the design of algorithm components must also incorporate software-engineering aspects. A key design goal is adaptability. This goal is important for maintenance throughout a project, prototypical development, and reuse in new, unforseen contex ...

Keywords: algorithm engineering

46 A system and language for building system-specific, static analyses

Seth Hallem, Benjamin Chelf, Yichen Xie, Dawson Engler

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(276.96 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a novel approach to bug-finding analysis and an implementation of that approach. Our goal is to find as many serious bugs as possible. To do so, we designed a flexible, easy-to-use extension language for specifying analyses and an efficent algorithm for executing these extensions. The language, *metal*, allows the users of our system to specify a broad class of analyses in terms that resemble the intuitive description of the rules that they check. The system, *xgcc* ...

Keywords: error detection, extensible compilation

47 The Pan language-based editing system for integrated development

Robert A. Ballance, Susan L. Graham, Michael L. Van De Vanter

October 1990 **ACM SIGSOFT Software Engineering Notes , Proceedings of the fourth ACM SIGSOFT symposium on Software development environments**, Volume 15 Issue 6

Full text available: pdf(2.11 MB) Additional Information: full citation, abstract, references, citings, index terms

Powerful editing systems for developing complex software documents are difficult to engineer. Besides requiring efficient incremental algorithms and complex data structures, such editors must integrate smoothly with the other tools in the environment, maintain a sharable database of information concerning the documents being edited, accommodate

flexible editing styles, provide a consistent, coherent, and empowering user interface, and support individual variations and project-wide configura ...

48 Toward automating the software-development cycle
Karen A. Frenkel
June 1985 **Communications of the ACM**, Volume 28 Issue 6
Full text available: pdf(1.24 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Knowledge-intensive rather than labor-intensive processes are being advanced to spur programming productivity.

49 Parlog86 and the dining logicians
G. A. Ringwood
January 1988 **Communications of the ACM**, Volume 31 Issue 1
Full text available: pdf(1.90 MB) Additional Information: full citation, abstract, references, citings, index terms, review

A classic problem in concurrent programming is that of the "dining philosophers" which challenges the power of any aspiring concurrent program language. Recently, a growing number of logic programming languages have been refined to handle concurrent programming, one in particular is Parlog86.

50 Program visualization: the art of mapping programs to pictures
Gruia-Catalin Roman, Kenneth C. Cox
June 1992 **Proceedings of the 14th international conference on Software engineering**
Full text available: pdf(1.10 MB) Additional Information: full citation, references, citings, index terms

51 Formal refinement patterns for goal-driven requirements elaboration
Robert Darimont, Axel van Lamsweerde
October 1996 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering**, Volume 21 Issue 6
Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, references, citings, index terms

Requirements engineering is concerned with the identification of high-level goals to be achieved by the system envisioned, the refinement of such goals, the operationalization of goals into services and constraints, and the assignment of responsibilities for the resulting requirements to agents such as humans, devices and programs. Goal refinement and operationalization is a complex process which is not well supported by current requirements engineering technology. Ideally some form of formal su ...

Keywords: design patterns, formal methods, goal-drive requirements engineering, refinement, reuse of specifications and proof

52 Operational specification languages
Pamela Zave
January 1983 **Proceedings of the 1983 annual conference on Computers : Extending the human resource**
Full text available: pdf(852.45 KB) Additional Information: full citation, abstract, references, index terms

The "operational approach" to software development is based on separation of problem-oriented and implementation-oriented concerns, and features executable specifications and transformational implementation. "Operational specification languages" are executable specification languages designed to fit the goals, assumptions, and strategies of the operational approach. This paper defines the operational approach and surveys the existing operational specification languag ...

53 Session 3B: Software testing: Fault localization using execution traces
Margaret Francel, Spencer Rugaber
April 1992 **Proceedings of the 30th annual Southeast regional conference**

Full text available: pdf(474.50 KB) Additional Information: full citation, abstract, references

This paper describes an approach to the automatic localization of program faults using execution traces. The traces are used to construct a directed graph that models the propagation of values during execution. Knowledge of which output values are incorrect is then used to narrow the region that must be examined by the person debugging the program. Algorithms to construct and analyze the graph have been designed and implemented.

54 Objects in concurrent logic programming languages
Kenneth Kahn, Eric Dean Tribble, Mark S. Miller, Daniel G. Bobrow
June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 21 Issue 11

Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent Prolog supports object-oriented programming with a clean semantics and additional programming constructs such as incomplete messages, unification, direct broadcasting, and concurrency synchronization [Shapiro 1983a]. While it provides excellent computational support, we claim it does not provide good notation for expressing the abstractions of object-oriented programming. We describe a preprocessor that remedies this problem. The resulting language, Vulcan, is then used as a behi ...

55 VLSI CAD tool integration using the Ulysses environment
Michael L. Bushnell, S. W. Director
July 1986 **Proceedings of the 23rd ACM/IEEE conference on Design automation**

Full text available: pdf(871.45 KB) Additional Information: full citation, abstract, references, citings, index terms

Ulysses is a VLSI CAD environment which effectively addresses the problems associated with CAD tool integration. Specifically, Ulysses allows the integration of CAD tools into a design automation system, the codification of a design methodology, and the representation of a design space. Ulysses keeps track of the progress of a design and allows exploration of the design space. The environment employs artificial intelligence techniques, functions as an inter ...

56 A review of automated debugging systems: knowledge, strategies and techniques
M. Ducassé, A.-M. Emde
April 1988 **Proceedings of the 10th international conference on Software engineering**

Full text available: pdf(982.37 KB) Additional Information: full citation, abstract, references, citings, index terms

Our review is based on descriptions of 18 existing automated systems on program debugging and of a dozen cognitive studies on debugging. We propose a classification of debugging knowledge, and a description of the corresponding knowledge representation in the systems. Then we propose a classification of global debugging strategies used in the systems, and a description of the corresponding techniques. We assess the identified strategies from a real world program development point of view.

57 The teachable language comprehender: a simulation program and theory of language
M. Ross Quillian
August 1969 **Communications of the ACM**, Volume 12 Issue 8

Full text available: pdf(2.39 MB) Additional Information: full citation, abstract, references, citings, index terms

The Teachable Language Comprehender (TLC) is a program designed to be capable of being taught to "comprehend" English text. When text which the program has not seen before is

input to it, it comprehends that text by correctly relating each (explicit or implicit) assertion of the new text to a large memory. This memory is a "semantic network" representing factual assertions about the world. The program also creates copies of the parts of its memory which ha ...

Keywords: computer linguistics, human memory simulation, linguistic performance theory, natural language comprehension, natural language processing, psychological simulation, teachable computer program

58 The new (1982) Computing Reviews classification system—final version
Jean E. Sammet, Anthony Ralston
January 1982 **Communications of the ACM**, Volume 25 Issue 1
Full text available: pdf(731.04 KB) Additional Information: full citation, citings, index terms

59 The relation between problems in large-scale concurrent systems and distributed databases
G. Agha
January 2000 **Proceedings of the first international symposium on Databases in parallel and distributed systems**
Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, index terms

We first describes the state of the art in models of concurrency. The models are analyzed along two dimensions: communication and computation. The paper then discusses some problems which make it difficult to realize large-scale concurrent systems. Such problems include compositionality, heterogeneity, debugging, resource management, and concurrency control. Some useful comparisons are drawn to problems in distributed databases and it is argued that solutions to these problems cross discipl ...

60 Breaking the complexity barrier again
Terry Winograd
November 1973 **Proceedings of the 1973 meeting on Programming languages and information retrieval**, Volume 9 , 10 Issue 3 , 1
Full text available: pdf(1.75 MB) Additional Information: full citation, references, citings

Terms used **backtracking history transformation tree result** Found **64** of **10,202**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 61 - 64 of 64 Result page: previous 1 2 3 **4**

Relevance scale

61 Workshop on object-oriented programming ECOOP 1987, Paris, June 18, 1987

January 1988 **ACM SIGPLAN Notices**, Volume 23 Issue 1

Full text available: pdf(1.22 MB) Additional Information: full citation, citings, index terms

62 Coca: an automated debugger for C

Mireille Ducassé

May 1999 **Proceedings of the 21st international conference on Software engineering**

Full text available: pdf(1.10 MB) Additional Information: full citation, references, citings, index terms

Keywords: automated debugging, debugging language, debugging tool, program behavior understanding, trace query mechanism

63 The Recovery Manager of the System R Database Manager

Jim Gray, Paul McJones, Mike Blasgen, Bruce Lindsay, Raymond Lorie, Tom Price, Franco Putzolu, Irving Traiger

June 1981 **ACM Computing Surveys (CSUR)**, Volume 13 Issue 2

Full text available: pdf(1.75 MB) Additional Information: full citation, references, citings, index terms

64 Technical contributions: Roster of programming languages for 1973

Jean E. Sammet

November 1974 **ACM SIGPLAN Notices**, Volume 9 Issue 11

Full text available: pdf(1.62 MB) Additional Information: full citation

Results 61 - 64 of 64 Result page: previous 1 2 3 **4**